WIS2 Training Cheat Sheet

The following cheat sheet will attempt to provide you with all the basic commands you may need when using the WIS2Box.

Student VM Access

You can access your dedicated VM on the local WIS2-training network using an **SSH-client** such as PuTTy.

- Host:
- Username:
- Password:

Docker

Using a New Image

- Building the image:

```
docker build -t <image_name> <dir_of_dockerfile>
```

Note: If you are in the directory of the Docker file already, this is more simply:

```
docker build -t <image_name> .
```

Volume Management

- Create a volume:

```
docker volume create <volume_name>
```

- List all created volumes:

```
docker volume ls
```

- Display detailed information on a volume:

```
docker volume inspect <volume_name>
```

- Remove a volume:

```
docker volume rm <volume_name>
```

- Remove all unused volumes:

```
docker volume prune
```

Container Management

- Create a container from an image, with an interactive terminal (`it`) and a mounted volume (`v`):

```
docker run -it -v ${pwd}:/app <image_name>
```

- Display a list of currently running containers:

```
docker ps
```

  ...or a list of all containers:

```
docker ps -a
```

- Start a stopped container:

```
docker start <container_name>
```

- Enter the interactive terminal of a running container:

```
docker exec -it <container_name> bash
```

- Remove a container

```
docker rm <container_name>
```

- Remove a running container:

```
docker rm -f <container_name>
```

Bash

Directory Navigation

- Entering a true directory:

```
cd /folder_1/folder_2
```

- Entering a local directory:

```
cd ./folder
```

- Move one directory upwards:

```
cd ..
```

- Move to the previously used directory:

```
cd -
```

File Management

- Listing files present in a directory:

```
ls
```

- Create a file:

```
touch <file_name>
```

- Copy one file to another:

```
cat <file_1> >> <file_2>
```

or

```
cp <file_1> <file_2>
```

- Delete a file:

```
rm <file_name>
```

- Delete all files with the same file extension:

```
rm *.<file_extension>
```

- Create a folder

```
mkdir <folder_name>
```

Connecting Commands

This routes the output of one command to another command, and is done using the pipe | symbol:

```
command_1 |command_2
```

- Restrict outputs to those containing keyword:

  ```
  command |grep <keyWord>
  ```

  - Ignoring case:

    ```
    command |grep -i <keyword>
    ```

  - Count matching lines:

    ```
    command |grep -c <keyWord>
    ```

  - Return outputs not containing keyword:

    ```
    command |grep -v <keyWord>
    ```

- Display output one screen at a time:

  ```
  command |more
  ```

  …with controls:

  - Scroll down line by line: *enter*
  - Go to next page: *space bar*
  - Go back one page: *b*

ecCodes Commands

- Display the data contained in a BUFR file:

  ```
  bufr_dump -p my_bufr
  ```

- Compare the differences between two BUFR files:

  ```
  bufr_compare <bufr_1> <bufr_2>
  ```

  - Ignore/blacklist keys from the comparison:

```
bufr_compare -b <key_1,key_2,key_3> <bufr_1> <bufr_2>
```

Performing Multiple Commands (One-Liners)

Multiple commands can be ran in sequential order from the same line using the semi-colon `;` symbol:

```
command_1; command_2; command_3
```

WIS2Box

Installing

- Build the WIS2Box:

  ```
  python3 wis2box-ctl.py build
  ```

- Update the WIS2Box:

  ```
  python3 wis2box-ctl.py update
  ```

- Start the WIS2Box:

  ```
  python3 wis2box-ctl.py start
  ```

- Login to the *wis2box-management* container:

  ```
  python3 wis2box-ctl.py login
  ```

- Verify all containers are running:

  ```
  python3 wis2box-ctl.py status
  ```

Metadata and Observations

- Publish discovery metadata:

```
wis2box metadata discovery publish <discovery_metadata_dir.yml>
```

- Add observation collections from discovery metadata:

```
wis2box data add-collection <discovery_metadata_dir.yml>
```

- Ingest data into the *wis2box-incoming* bucket:

```
wis2box data ingest --topic-hierarchy <topic.hierarchy> --path
↪  <observation_dir>
```

- Publish stations:

```
wis2box metadata station publish-collection
```

SYNOP2BUFR

- Convert a SYNOP message to BUFR:

```
synop2bufr transform --metadata <my_file.csv> --output-dir
↪  <./my_folder> --year <message_year> --month <message_month>
↪  <SYNOP_file_dir.txt>
```

Note: The options for this command are not required, and if not specified take the following default values:

Option	Default
–metadata	metadata.csv
–output-dir	The current working directory.
–year	The current year.
–month	The current month.

CSV2BUFR

- Create a template mappings file:

```
csv2bufr mappings create <BUFR descriptors> --output <output_dir>
```

- Convert a CSV file to BUFR:

```
csv2bufr data transform --bufr-template <my_template.json> --output-dir
↪  <./my_folder> <CSV_file_dir.csv>
```